Stochastic Variational Inference

MPRI — Cours 2.40 “Probabilistic Programming Languages”

Xavier Rival

INRIA, ENS, CNRS

Oct, 24th. 2024

Probabilistic Programming Languages Inference

Inference

Evaluation of the posterior probability distribution

Caveats:

- exact inference either **impossible** or **not tractable** in general
- **many techniques** for approximate inference
  several of them discussed in the previous classes

Today's plan: **stochastic variational inference (SVI)**

- intuition and definition
  how to formalize the semantics of SVI
- issues: it may not work, **unless some conditions are satisfied**
  questions: how to characterize these conditions, how to check them?

We consider **Pyro**, a probabilistic programming language implemented over Python with support for SVI, available at `https://pyro.ai`.
Formalization with a subset of the course language

Outline

Evaluation of probabilistic programs

What does it mean to evaluate a probabilstic program ?
How to do that ?

Enumeration:

- run all the executions and compute the probability of each of them
- only works in the finite case...

Importance sampling:

- compute a family of executions together with their probability
- executions are chosen at random (hopefully well)

Rejection sampling:

- compute a family of executions
- reject unlikely executions, accumulate those with high probability

Variational inference:

- search for a simpler program that is close enough
- ... more on this today...

A first, very basic model

We build a model step by step, to construct an example for SVI.

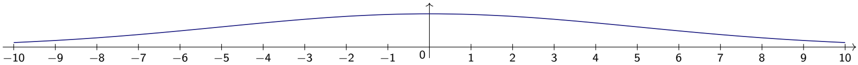
Model Pyro code:

```
def model():
  x = pyro.sample("v", Normal(0., 5.))
```

Meaning:

- `sample`: draws a value based on a distribution
  in this case, normal distribution, mean 0, standard deviation 5
- *i.e.*, values of variable random `v` (python variable `x`) distributed around 0
  with some imprecision

Distribution over executions based on the final value of `x`:



A second, more interesting model

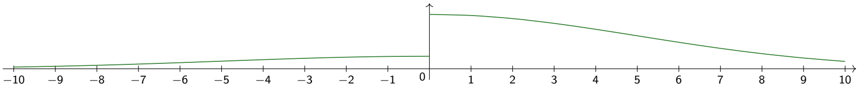
Model Pyro code:

```
def model():
  x = pyro.sample("v", Normal(0., 5.))
  if (x > 0):
    pyro.sample("obs", Normal(1., 1.), obs=x)
  else:
    pyro.sample("obs", Normal(-2., 1.), obs=x)
```

Meaning:

- `sample` without `obs=...`: sampling, as before
- `sample` with `obs=...`: conditioning determined by observation

Distribution on random variable `v`:

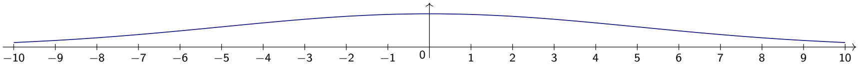


Distribution defined by the model

Model Pyro code:

```
def model():
  x = pyro.sample("v", Normal(0., 5.))
  if (x > 0):
    pyro.sample("obs", Normal(1., 1.), obs=x)
  else:
    pyro.sample("obs", Normal(−2., 1.), obs=x)
```

Prior on random variable `v`, before observation taken into account:
i.e., when observations on the value of `obs` are ignored

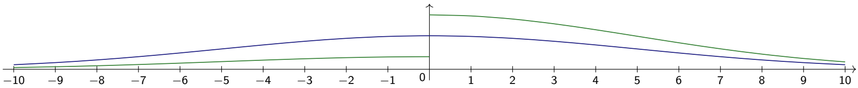


Distribution defined by the model

Model Pyro code:

```
def model():
  x = pyro.sample("v", Normal(0., 5.))
  if (x > 0):
    pyro.sample("obs", Normal(1., 1.), obs=x)
  else:
    pyro.sample("obs", Normal(-2., 1.), obs=x)
```

Posterior distribution on random variable v, after observations on obs and compared with the prior:

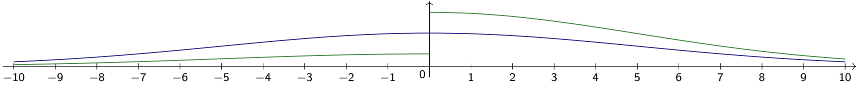


Distribution defined by the model

Model Pyro code:

```
def model():
  x = pyro.sample("v", Normal(0., 5.))
  if (x > 0):
    pyro.sample("obs", Normal(1., 1.), obs=x)
  else:
    pyro.sample("obs", Normal(−2., 1.), obs=x)
```

Posterior distribution on random variable v, after observations on obs and compared with the prior:



Can we **discover a simpler, accurate enough approximation of the posterior, as a program ?**

Model approximation with a parameterized "guide"

SVI main Idea:
specify a template for a family of candidate functions to approximate the posterior, then choose among them the most suitable one

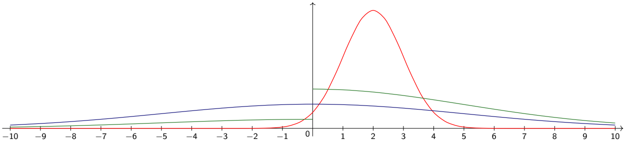
Guide

Companion program **with randomized parameter**, aimed at **approximating the posterior distribution** defined in the model

In our example: **sampling the parameter** from a **normal distribution**

```
def guide():
  xtheta = pyro.param("theta", 3.)
  x = pyro.sample("v", Normal(xtheta, 1.))
```

One instance of the guide, with a positive θ (**expected outcome**)



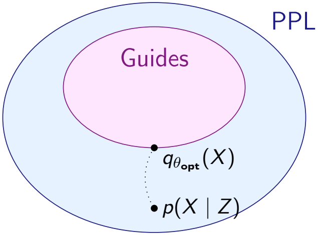
Objective of variational inference

SVI first question

How to define the optimization objective ? How to define the best guide ?

Data: probablistic program P called **model**
samples variables X, **observes variables** Z

- We fix the family of guides
  programs with **special variables identified as parameters**
- Probabilistic programs denote **posterior probability distributions**
  - $p(X|Z)$ for model
  - $q_\theta(X)$ for guide instance θ
- We fix a **distance over distributions** d
  in general, KL divergence



Objective: guide parameter θ_{opt} instance that minimizes distance

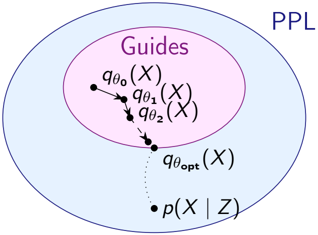
Stochastic computation of an approximation of the objective

SVI second question

How to solve the optimization problem in practice ?

Gradient descent algorithm:

1. Pick initial parameter θ_0
2. **Estimate the gradient** ∇_θ of the distance at point θ_0 and make a step in that direction, to compute θ_1
3. Repeat to compute $\theta_2, \ldots$ and **stop after** K **steps**



Stochastic approximation of the gradient at θ:

- Sample N runs, based on the guide
- Compute an estimate gradient based on the average variation

Convergence: based on properties of the optimization problem

Outline

Towards a stochastic approximation of the distance

Main step of the gradient descent algorithm:
$\Mapsto$ **estimation of the distance between distributions**

High-level overview of a single step of gradient descent (full algorithm shown later):

1. **sample** a number of executions (stochastic)
2. for each sampled execution, **compute probability density** (for short, density)
3. **produce a gradient approximation** based on the executions and their densities

How to compute the probability density of an expression ?

In the following, we propose a **density semantics** that answers the question:

- first, we recall the **kernel semantics**:
  maps a state into a probability distribution over outcomes
- then, we construct a new semantics called **density semantics**, that produces not only output states but also probability density

Implementation: an evaluator following this density semantics (instrumented runtime)...

A basic imperative probabilistic programming language

We define a **minimalistic form** of ByoPPL.

A few assumptions:

- **imperative** control structures (while language),
- **real** numbers (not floating point)
- only normal distributions in the minimalistic syntax below
  though examples may use other distributions as well
- **countable set of random variables**, represented with **string** names

Basic syntax:

$$
\begin{array}{rcl}
E, B, S & & \text{real, boolean, string expressions} \\
C & ::= & \text{commands} \\
& | & \mathrm{skip} \mid C_0; C_1 \mid x := E \\
& | & \mathrm{if}\, B\, \{C\}\, \mathrm{else}\, \{C\} \mid \mathrm{while}\, B\, \{C\} \\
& | & x := \mathrm{sample}_{\mathcal{N}}(k, E_0, E_1) \\
& & \quad k: \text{ random variable name, } E_0: \text{ mean, } E_1: \text{ standard dev.} \\
& | & \mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2) \\
& & \quad E_0: \text{ observed value, } E_1: \text{ mean, } E_2: \text{ standard dev.}
\end{array}
$$

Towards two semantics

Common characteristics:

- executions **record all random choices**
  random choices are **stored in a dictionary binding names to values**
- states comprise a **memory** and a **random dictionary**

Example:

$$x := \mathrm{sample}_{\mathcal{N}}(\mathtt{s0}, 0, 1);$$
$$y := \mathrm{sample}_{\mathcal{N}}(\mathtt{s1}, 2, 1);$$

initial state: $(\mathtt{s2} \mapsto 2), (x \mapsto 8, y \mapsto 9)$
final state: $(\mathtt{s0} \mapsto 0.5, \mathtt{s1} \mapsto 1.5, \mathtt{s2} \mapsto 2), (x \mapsto 0.5, y \mapsto 1.5)$

- consequence 1: a given random variable is sampled at most once
- consequence 2: initially present unsampled variables remain in the dictionary
- consequence 3: sampling from a variable already in the dictionary is blocking

Our two semantics:

1. **Kernel semantics**: maps states into probability distributions over states
2. **Density semantics**: maps states into states + probability density

Notations

We fix the following definitions:

- $\mathbb{X}$: set of **program variables**
- $\mathbb{V}$: set of **scalar values** (we assume real numbers, not floating point)
- $\mathbb{M} = [\mathbb{X} \to \mathbb{V}]$: set of **memory states**
  (notation: $\mu \in \mathbb{M}$)
- $\mathbb{K}$: set of **random variables**, corresponding to strings
- $\mathbb{P} = [\mathbb{K} \to \mathbb{V}]$: set of **random databases** (or random dictionaries)
  (notation: $\rho \in \mathbb{P}$)
- $\mathbb{S} = \mathbb{M} \times \mathbb{P}$: set of **states**
  (notation: $\sigma \in \mathbb{S}$)
- given a measurable set $\mathbb{A}$, we note $\mathcal{M}(\mathbb{A})$ for the set of measurable subsets of $\mathbb{A}$

Assumptions

We fix the following assumptions:

- usual structure of **measurable space structures** over $\mathbb{M}$, $\mathbb{P}$, and $\mathbb{S}$
- notion of **probability kernel** over $\mathbb{A}$ and measurable set $\mathbb{A}'$:
  function from $\mathbb{A}$ to probability distributions over $\mathbb{A}'$ (previous lecture)
  - probability kernel: total measure is 1, noted $\mathcal{K}(\mathbb{A}, \mathbb{A}')$
  - sub-probability kernel: total measure ≤ 1, noted $\mathcal{K}(\mathbb{A}, \mathbb{A}')_{\mathsf{sub}}$

Semantics:

- we assume a semantics of expressions:
  for all express E:
  $$[\![E]\!] : \mathbb{M} \to \mathbb{V}$$
  (random variables not used)
- Semantics of commands:
  $$[\![C]\!]_{\mathcal{M}} \in \mathcal{K}(\mathbb{S}, \mathbb{S} \times \mathbb{R}^{+})_{\mathsf{sub}}$$

Or equivalently:

$$[\![C]\!]_{\mathcal{M}} : \mathbb{S} \to (\mathcal{M}(\mathbb{S} \times \mathbb{R}^{+}) \to_{\mathcal{M}} [0, 1]) \equiv (\mathbb{M} \times \mathbb{P}) \to \mathcal{M}(\mathbb{M} \times \mathbb{P} \times \mathbb{R}^{+}) \to_{\mathcal{M}} [0, 1]$$

Measure semantics: signature and a few cases

$$[\![C]\!]_{\mathcal{M}} : \mathbb{S} \to (\mathcal{M}(\mathbb{S} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1]) \equiv (\mathbb{M} \times \mathbb{P}) \to \mathcal{M}(\mathbb{M} \times \mathbb{P} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1]$$

Assignment statement $x := E$

$$[\![x := E]\!]_{\mathcal{M}}(\mu, \rho)(S) \triangleq \mathbb{1}_{[((\mu[x \mapsto [\![E]\!](\mu)], \rho), 1) \in S]}$$

- weight is not modified
- variable x is updated in the store
- note: expressions should not read random variables directly

Measure semantics: signature and a few cases

$$[\![C]\!]_{\mathcal{M}} : \mathbb{S} \to (\mathcal{M}(\mathbb{S} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1]) \equiv (\mathbb{M} \times \mathbb{P}) \to \mathcal{M}(\mathbb{M} \times \mathbb{P} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1]$$

Assignment statement $x := E$

Sample statement $\mathrm{sample}_{\mathcal{N}}(s, E_0, E_1)$

$$\begin{array}{l}[\![x := \mathrm{sample}_{\mathcal{N}}(k, E_1, E_2)]\!]_{\mathcal{M}}(\mu, \rho)(S) \triangleq \\ \quad \mathbb{1}_{[k \notin \mathrm{Dom}(\rho)]} \cdot \mathbb{1}_{[[\![E_2]\!](\mu) \in \mathbb{R}^{+*}]} \\ \quad \cdot \int \mathrm{d}v \left(\mathrm{pdf}_{\mathcal{N}}(v; [\![E_1]\!](\mu), [\![E_2]\!](\mu)) \cdot \mathbb{1}_{[(\mu[x \mapsto v], \rho[k \mapsto v], 1) \in S]}\right)\end{array}$$

- crashes when sampling from a rand. var. not in the random database
- crashes when standard deviation is negative
- otherwise updates the states and rdb with the new sample,
  integrate over the density of the sampled distribution
  and do this for all possible samples (hence the sum)

Notation: $\mathrm{pdf}_{\mathcal{N}}(v; m, d)$: **probability density** at v of the normal distribution of mean m and standard deviation s

Measure semantics: signature and a few cases

$$[\![C]\!]_{\mathcal{M}} : \mathbb{S} \to (\mathcal{M}(\mathbb{S} \times \mathbb{R}^{+}) \to_{\mathcal{M}} [0,1]) \equiv (\mathbb{M} \times \mathbb{P}) \to \mathcal{M}(\mathbb{M} \times \mathbb{P} \times \mathbb{R}^{+}) \to_{\mathcal{M}} [0,1]$$

Assignment statement $x := E$

Sample statement $\mathrm{sample}_{\mathcal{N}}(s, E_0, E_1)$

Score statement $\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)$

$$[\![\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)]\!]_{\mathcal{M}}(\mu, \rho)(S) \triangleq \mathbb{1}_{[[\![E_2]\!](\mu) \in \mathbb{R}^{+*}]} \cdot \mathbb{1}_{[((\mu,\rho), \mathsf{pdf}_{\mathcal{N}}([\![E_0]\!](\mu); [\![E_1]\!](\mu), [\![E_2]\!](\mu))) \in S]}$$

- crashes when standard deviation is negative
- otherwise state left unmodified
  score the density of the distribution for the observed value

Measure semantics: signature and a few cases

$$[\![C]\!]_{\mathcal{M}} : \mathbb{S} \to (\mathcal{M}(\mathbb{S} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1]) \equiv (\mathbb{M} \times \mathbb{P}) \to \mathcal{M}(\mathbb{M} \times \mathbb{P} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1]$$

Assignment statement $x := E$

Sample statement $\mathrm{sample}_{\mathcal{N}}(s, E_0, E_1)$

Score statement $\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)$

For all command C, $[\![C]\!]_{\mathcal{M}}$ is measurable
and defines a sub-probability kernel from $\mathbb{S}$ to $\mathbb{S} \times \mathbb{R}^+$

Measure semantics (or kernel semantics)

$$
\begin{aligned}
&[\![\mathrm{skip}]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq \mathbb{1}_{[((\mu,\rho),1)\in S]} \\
&[\![x := E]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq \mathbb{1}_{[((\mu[x\mapsto [\![E]\!](\mu)],\rho),1)\in S]} \\
&[\![C_0; C_1]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq \int [\![C_0]\!]_{\mathcal{M}}(\mu,\rho)(\mathrm{d}(\sigma_0,w_0)) \int [\![C_1]\!]_{\mathcal{M}}(\sigma_0)(\mathrm{d}(\sigma_1,w_1)) \mathbb{1}_{[(\sigma_1, w_0 w_1)\in S]} \\
&[\![\mathrm{if}\ B\ \{C_0\}\mathrm{else}\{C_1\}]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{true}]} \cdot [\![C_0]\!]_{\mathcal{M}}(\mu,\rho)(S) + \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{false}]} \cdot [\![C_1]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&[\![\mathrm{while}\ B\ \{C\}]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq (\mathsf{Fix} F)(\mu,\rho)(S) \\
&\quad \text{where } F(\phi)(\mu,\rho)(S) = \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{false}]} \cdot \mathbb{1}_{[((\mu,\rho),1)\in S]} \\
&\qquad\qquad + \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{true}]} \cdot \int [\![C]\!]_{\mathcal{M}}(\mu,\rho)(\mathrm{d}(\sigma_0,w_0)) \int \phi(\sigma_0)(\mathrm{d}(\sigma_1,w_1)) \mathbb{1}_{[(\sigma_1,w_0 w_1)\in S]} \\
&[\![x := \mathrm{sample}_{\mathcal{N}}(k, E_1, E_2)]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq \mathbb{1}_{[k\notin \mathrm{Dom}(\rho)]} \cdot \mathbb{1}_{[[\![E_2]\!](\mu)\in\mathbb{R}^{+*}]} \\
&\qquad\qquad \cdot \int \mathrm{d}v \left(\mathsf{pdf}_{\mathcal{N}}(v; [\![E_1]\!](\mu), [\![E_2]\!](\mu)) \cdot \mathbb{1}_{[(\mu[x\mapsto v],\rho[k\mapsto v],1)\in S]}\right) \\
&[\![\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)]\!]_{\mathcal{M}}(\mu,\rho)(S) \\
&\qquad \triangleq \mathbb{1}_{[[\![E_2]\!](\mu)\in\mathbb{R}^{+*}]} \cdot \mathbb{1}_{[((\mu,\rho),\mathsf{pdf}_{\mathcal{N}}([\![E_0]\!](\mu);[\![E_1]\!](\mu),[\![E_2]\!](\mu)))\in S]}
\end{aligned}
$$

A basic example

We consider the program: $C \triangleq \begin{cases} x := \mathrm{sample}(a, 0, 5); \\ \mathrm{observe}(x, 3, 1); \end{cases}$

- prior: x close to 0, low confidence
- posterior: noisy observation that x is close to 3

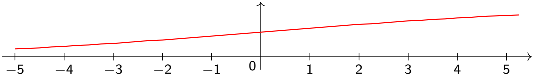
Measure semantics, starting from $\mu_I = \{x \mapsto ?\}$ and $\rho_I = \emptyset$,

$[\![C]\!]_{\mathcal{M}}(\mu_I, \rho_I)(S) = [\![C]\!]_{\mathcal{M}}(\mu_I, \rho_I)(S \cap \{(\{x \mapsto v\}, \{a \mapsto v\}, \mathrm{pdf}_{\mathcal{N}}(3; v, 1)) \mid v \in \mathbb{R}\})$

(*i.e.*, other ouptut state, density pairs do not count)

Cumulated measure, *i.e.*, over $\{(\{x \mapsto v\}, \{a \mapsto v\}, \mathrm{pdf}_{\mathcal{N}}(3; v, 1)) \mid v \leq \alpha\}$

$$\int_{-\infty}^{\alpha} [\![C]\!]_{\mathcal{M}}(\mu_I, \rho_I)(\{(\{x \mapsto v\}, \{a \mapsto v\}, \mathrm{pdf}_{\mathcal{N}}(3; v, 1)) \mid v \in \mathbb{R}\})\mathrm{d}v$$



Issue: per execution weight and probability over executions remain separate

Outline

Towards a density semantics

For the **definition of SVI**, the measure semantics has several **limitations**:

- the weight of executions ($\mathrm{observe}$ commands) and the measure over output configurations are computed separately
- as mentioned earlier, we look for a way to compute the probability density of each execution together with its output

Important notes:

- as we seek for a semantics that maps an initial state to an output configuration, we need to assume random samples fixed beforehand and are **consumed** during the execution
  i.e., the ρ input collects values to be sampled, each sample pops a value
  note this is the converse of the measure semantics convention
- we also expect this semantics to be **deterministic**, *i.e.*, for any given initial state, the execution of a command should produce a single output configuration
- executions **may not terminate or may crash** so a special output configuration is needed
  i.e., $\perp$ denotes executions that either fail or do not terminate

Density semantics basic definition

We recall the form of the measure semantics:

$$[\![C]\!]_{\mathcal{M}} : \mathbb{S} \to (\mathcal{M}(\mathbb{S} \times \mathbb{R}^+) \to_{\mathcal{M}} [0,1])$$

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \to_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^+ \times \mathbb{R}^+ \uplus \{\perp\}$$

When $[\![C]\!]_{\mathcal{D}}(\mu, \rho) = (\mu', \rho', w', p')$:

- μ' denotes the new memory
- ρ' denotes the remaining part of the random dictionary
- w' denotes the execution weight (scoring)
- p' denotes its probability density

Definition of the semantics: **exercise**!

Density semantics of the skip command

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \rightarrow_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$$

Command C:

$$\text{skip}$$

Measure semantics:

$$[\![\text{skip}]\!]_{\mathcal{M}}(\mu, \rho)(S) = \mathbb{1}_{[((\mu,\rho),1)\in S]}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**

Density semantics of assignment commands

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \rightarrow_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$$

Command C:

$$x := E$$

Measure semantics:

$$[\![x := E]\!]_{\mathcal{M}}(\mu, \rho)(S) = \mathbb{1}_{[((\mu[x \mapsto [\![E]\!](\mu)], \rho), 1) \in S]}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**
   under the assumption that expressions never crash
2. What would happen if we assume expressions may crash

Density semantics of sequences

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \rightarrow_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$$

Command C:

$$C_0; C_1$$

Measure semantics:

$$\begin{aligned} & [\![C_0; C_1]\!]_{\mathcal{M}}(\mu, \rho)(S) = \\ & \qquad \int [\![C_0]\!]_{\mathcal{M}}(\mu, \rho)(\mathrm{d}(\sigma_0, w_0)) \int [\![C_1]\!]_{\mathcal{M}}(\sigma_0)(\mathrm{d}(\sigma_1, w_1)) \mathbb{1}_{[(\sigma_1, w_0 w_1) \in S]} \end{aligned}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**
2. What happens with composition ?
   Try to imagine a "lift" operator that makes $[\![.]\!]_{\mathcal{D}}$ easier to compose and update the definition for $[\![.]\!]_{\mathcal{D}}$ accordingly

Density semantics of condition tests

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \rightarrow_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$$

Command C:

$$\text{if } B \,\{C_0\}\text{else}\{C_1\}$$

Measure semantics:

$$\begin{array}{l} [\![\text{if } B \,\{C_0\}\text{else}\{C_1\}]\!]_{\mathcal{M}}(\mu, \rho)(S) = \\ \qquad \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{true}]} \cdot [\![C_0]\!]_{\mathcal{M}}(\mu, \rho)(S) + \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{false}]} \cdot [\![C_1]\!]_{\mathcal{M}}(\mu, \rho)(S) \end{array}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**

Density semantics of loop commands

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \rightarrow_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^+ \times \mathbb{R}^+ \uplus \{\perp\}$$

Command C:

$$\mathrm{while}\, B\, \{C\}$$

Measure semantics:

$$[\![\mathrm{while}\, B\, \{C\}]\!]_{\mathcal{M}}(\mu, \rho)(S) = (\mathsf{Fix} F)(\mu, \rho)(S)$$

where

$$\begin{aligned} F(\phi)(\mu, \rho)(S) = \; & \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{false}]} \cdot \mathbb{1}_{[((\mu,\rho),1)\in S]} \\ & + \mathbb{1}_{[[\![B]\!](\mu)=\mathsf{true}]} \cdot \int [\![C]\!]_{\mathcal{M}}(\mu, \rho)(\mathrm{d}(\sigma_0, w_0)) \int \phi(\sigma_0)(\mathrm{d}(\sigma_1, w_1)) \mathbb{1}_{[(\sigma_1, w_0 w_1)\in S]} \end{aligned}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**
2. Explain the assumptions required for the definition

Density semantics of sample commands

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \to_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$$

Command C:

$$x := \mathrm{sample}_{\mathcal{N}}(k, E_1, E_2)$$

Measure semantics:

$$\begin{array}{l} [\![x := \mathrm{sample}_{\mathcal{N}}(k, E_1, E_2)]\!]_{\mathcal{M}}(\mu, \rho)(S) = \\ \quad \mathbb{1}_{[k \notin \mathrm{Dom}(\rho)]} \cdot \mathbb{1}_{[[\![E_2]\!](\mu) \in \mathbb{R}^{+*}]} \\ \quad\quad \cdot \int \mathrm{d}v \left(\mathrm{pdf}_{\mathcal{N}}(v; [\![E_1]\!](\mu), [\![E_2]\!](\mu)) \cdot \mathbb{1}_{[(\mu[x \mapsto v], \rho[k \mapsto v], 1) \in S]} \right) \end{array}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**
2. Comment on error cases

Density semantics of observe commands

Signature of the density semantics

$$[\![C]\!]_{\mathcal{D}} : \mathbb{S} \rightarrow_{\mathcal{M}} \mathbb{S} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$$

Command C:

$$\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)$$

Measure semantics:

$$[\![\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)]\!]_{\mathcal{M}}(\mu, \rho)(S) = \mathbb{1}_{[[\![E_2]\!](\mu) \in \mathbb{R}^{+*}]} \cdot \mathbb{1}_{[((\mu,\rho), \mathrm{pdf}_{\mathcal{N}}([\![E_0]\!](\mu); [\![E_1]\!](\mu), [\![E_2]\!](\mu))) \in S]}$$

Exercise

1. **Propose a definition for $[\![.]\!]_{\mathcal{D}}$**
2. Comment on error cases

Density semantics

Lifting of semantic function $g : \mathbb{M} \times \mathbb{P} \rightarrow_{\mathcal{M}} \mathbb{M} \times \mathbb{P} \times \mathbb{R}^{+} \times \mathbb{R}^{+} \uplus \{\perp\}$, for **composition**:

$$\begin{array}{rcl} g^{\bullet}(\perp) & = & \perp \\ g^{\bullet}(\mu, \rho, w, p) & = & \left\{ \begin{array}{ll} \perp & \text{if } g(\mu, \rho) = \perp \\ (\mu', \rho', w \cdot w', p \cdot p') & \text{if} g(\mu, \rho) = (\mu', \rho', w', p') \end{array} \right. \end{array}$$

Semantics:

$$\begin{array}{rcl} [\![\mathrm{skip}]\!]_{\mathcal{D}}(\mu, \rho) & = & (\mu, \rho, 1, 1) \\ [\![x := E]\!]_{\mathcal{D}}(\mu, \rho) & = & (\mu[x \mapsto [\![E]\!](\mu)], \rho, 1, 1) \\ [\![C_0; C_1]\!]_{\mathcal{D}}(\mu, \rho) & = & [\![C_0]\!]_{\mathcal{D}}{}^{\bullet} \circ [\![C_1]\!]_{\mathcal{D}}(\mu, \rho) \\ [\![\mathrm{if}\ B\ \{C_0\}\mathrm{else}\{C_1\}]\!]_{\mathcal{D}}(\mu, \rho) & = & \text{if } [\![B]\!](\mu) = \text{true then } [\![C_0]\!]_{\mathcal{D}}(\mu, \rho) \text{ else } [\![C_1]\!]_{\mathcal{D}}(\mu, \rho) \\ [\![\mathrm{while}\ B\ \{C\}]\!]_{\mathcal{D}}(\mu, \rho) & = & (\text{Fix} G)(\mu, \rho) \end{array}$$

where $G(\psi)(\mu, \rho) = \text{if } [\![B]\!](\mu) = \text{false then } (\mu, \rho, 1, 1) \text{ else } \psi^{\bullet} \circ [\![C]\!]_{\mathcal{D}}(\mu, \rho)$

$$\begin{array}{l} [\![x := \mathrm{sample}_{\mathcal{N}}(k, E_1, E_2)]\!]_{\mathcal{D}}(\mu, \rho) \\ \quad = \left\{ \begin{array}{l} \text{if } k \notin \mathrm{Dom}(\rho) \vee [\![E_2]\!](\mu) \notin \mathbb{R}^{+*} \text{ then } \perp \\ \text{else } (\mu[x \mapsto \rho(k)],\ \rho \setminus \{k\},\ \text{pdf}_{\mathcal{N}}(\rho(k); [\![E_1]\!](\mu), [\![E_2]\!](\mu))) \end{array} \right. \end{array}$$

$$\begin{array}{l} [\![\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)]\!]_{\mathcal{D}}(\mu, \rho) \\ \quad = \left\{ \begin{array}{l} \text{if } ([\![E_2]\!](\mu) \notin \mathbb{R}^{+*}) \text{ then } \perp \\ \text{else } (\mu, \rho, \text{pdf}_{\mathcal{N}}([\![E_0]\!](\mu); [\![E_1]\!](\mu), [\![E_2]\!](\mu)), 1) \end{array} \right. \end{array}$$

Density of an execution and example

Definition: execution probability density

When $[\![C]\!]_{\mathcal{D}}(\mu_I, \rho_I) = (\mu, \emptyset, w, p)$, we let:

$$\mathcal{D}[\![C]\!](\mu_I, \rho_I) = w \cdot p$$

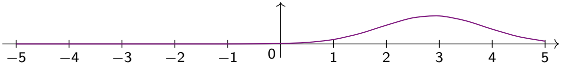
We consider again the program: $C \triangleq \left\{ \begin{array}{l} x := \mathrm{sample}(a, 0, 5); \\ \mathrm{observe}(x, 3, 1); \end{array} \right.$

- prior: x close to 0, low confidence
- posterior: noisy observation that x is close to 3

Semantics derived by simple calculation, starting from $\mu_I = \{x \mapsto ?\}$ and $\rho_I(v) = \{a \mapsto v\}$,

$$[\![C]\!]_{\mathcal{D}}((\mu_I, \rho_I), 1, 1) = ((\{x \mapsto v\}, \emptyset), \mathrm{pdf}_{\mathcal{N}}(3; v, 1), \mathrm{pdf}_{\mathcal{N}}(v; 0, 5))$$

Overall weighted density: $v \longmapsto \mathcal{D}[\![C]\!](\mu_i, \rho_i(v)) = \mathrm{pdf}_{\mathcal{N}}(3; v, 1) \cdot \mathrm{pdf}_{\mathcal{N}}(v; 0, 5)$



Density of an execution and example

Definition: execution probability density

When $[\![C]\!]_{\mathcal{D}}(\mu_I, \rho_I) = (\mu, \emptyset, w, p)$, we let:

$$\mathcal{D}[\![C]\!](\mu_I, \rho_I) = w \cdot p$$

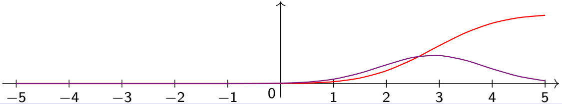
We consider again the program: $C \triangleq \begin{cases} x := \mathrm{sample}(a, 0, 5); \\ \mathrm{observe}(x, 3, 1); \end{cases}$

- prior: x close to 0, low confidence
- posterior: noisy observation that x is close to 3

Semantics derived by simple calculation, starting from $\mu_I = \{x \mapsto ?\}$ and $\rho_I(v) = \{a \mapsto v\}$,

$$[\![C]\!]_{\mathcal{D}}((\mu_I, \rho_I), 1, 1) = ((\{x \mapsto v\}, \emptyset), \mathrm{pdf}_{\mathcal{N}}(3; v, 1), \mathrm{pdf}_{\mathcal{N}}(v; 0, 5))$$

Cumulated weighted density: $v \longmapsto \int_{-\infty}^{v} \mathcal{D}[\![C]\!](\mu_i, \rho_i(x))\mathrm{d}x$



Link between density semantics and measure semantics

Definition of density: when $[\![C]\!]_{\mathcal{D}}(\mu_I, \rho_I) = (\mu, \emptyset, w, p)$, we have $\mathcal{D}[\![C]\!](\mu_I, \rho_I) = w \cdot p$

Theorem

Given a set of random dictonaries $P \subset \mathbb{P}$, we can measure the probability to evaluate exactly a $\rho \in P$ with either semantics, in an equivalent manner:

$$\begin{array}{rcl} \mathcal{M}(C, P) & \triangleq & \int (\mathbb{1}_{[\rho \in P]} \cdot |\mathrm{d}\rho| \cdot \mathcal{D}[\![C]\!](\mu_I, \rho)) \\ & = & \int [\![C]\!]_{\mathcal{M}}(\mu_I, \emptyset)(\mathrm{d}(\mu, \rho, w)) \cdot (w \cdot \mathbb{1}_{[\rho \in P]}) \end{array}$$

Proof: **exercise!**

Outline

KL divergence

Status so far:

- given model P and parameterized guide Q_θ, we have defined the distributions p and q_θ that they induce over random data-bases
- we next need to establish a **measure of dissimilarity** between p and q_θ, to be able to define the optimization objective of SVI

Definition: KL divergence (Kullback-Leibler divergence)

Given two probability distributions p_0, p_1 over the same measurable set $\mathcal{X}$, their **KL divergence** writes down as:

$$\mathbf{D}_{\mathrm{KL}}(p_0||p_1) = \mathbb{E}_{p_0}\left(\log \frac{p_0}{p_1}\right) = \int_{\mathcal{X}} p_0(x) \log \frac{p_0(x)}{p_1(x)} \mathrm{d}x$$

In the discrete case:

$$\mathbf{D}_{\mathrm{KL}}(p_0||p_1) = \sum_{x \in \mathcal{X}} p_0(x) \log \frac{p_0(x)}{p_1(x)}$$

Some properties of KL divergence

Main properties, for a given measurable space with measure $|.|$:

1. it is **positive**: for all p, q, $\mathbf{D}_{\mathrm{KL}}(p||q) \geq 0$
2. it is **null if and only if its arguments are equal almost everywhere**:

$$\mathbf{D}_{\mathrm{KL}}(p||q) = 0 \iff |x| \neq 0 \Longrightarrow p(x) = q(x)$$

3. it is **not symmetric**, which means that in general $\mathbf{D}_{\mathrm{KL}}(p||q) \neq \mathbf{D}_{\mathrm{KL}}(q||p)$
4. it **does not satisfy the triangular inequality**, which means that, in general, the inequality $\mathbf{D}_{\mathrm{KL}}(p_0||p_2) \leq \mathbf{D}_{\mathrm{KL}}(p_0||p_1) + \mathbf{D}_{\mathrm{KL}}(p_1||p2)$ does not hold

Due to the last two points it is called *divergence*, and not **distance**

Examples, over $X = \{0, 1\}$:

- p_0 defined by $p_0(0) = \frac{1}{2}$ and $p_0(1) = \frac{1}{2}$
- p_1 defined by $p_1(0) = \frac{1}{5}$ and $p_1(1) = \frac{4}{5}$
- p_2 defined by $p_2(0) = \frac{1}{10}$ and $p_2(1) = \frac{9}{10}$

Then:

- $\mathbf{D}_{\mathrm{KL}}(p_0||p_1) \approx 0.22$ but $\mathbf{D}_{\mathrm{KL}}(p_1||p_0) \approx 0.19$
- $\mathbf{D}_{\mathrm{KL}}(p_0||p_2) \approx 0.51$ but $\mathbf{D}_{\mathrm{KL}}(p_0||p_1) + \mathbf{D}_{\mathrm{KL}}(p_1||p_2) \approx 0.27$

SVI objective

Setup:

- model P; defines a probability distribution over sampled and observed random variables noted $p(z, x)$ where:
  z: sampled random variables (ultimately represented in ρ)
  x: observed random variables (ultimately not represented in ρ)
- guide Q, with a set of variables identified as parameters and noted t; given a value assignment $t \mapsto \theta$, defines a probability distribution $q_\theta(z)$ over sampled random variables

We defined: $\mathcal{M}(p, S) = \int \rho(\mathrm{d}\rho)\,(\mathbb{1}_{[\rho \in S]} \cdot \mathcal{D}[\![p]\!](\mu_I, \rho))$

It can be **normalized** into a **probability measure** iff $\mathcal{M}(p, \mathbb{P}) \in \mathbb{R}^{+*}$

Objective:

- beforehand, fix the family of programs q_θ as potential approximants of C
  1. with $\mathcal{M}(q_\theta, \mathbb{P}) = 1$, which is ensured **if q_θ always terminates**
  2. with density 1, which is ensured **if it contains no occurrence of** observe
- compute optimal θ to minimize

$$\mathbf{D}_{\mathrm{KL}}(\mathcal{D}[\![q_\theta]\!](\mu_I, \rho), \mathcal{D}[\![C]\!](\mu_I, \rho))$$

SVI optimization objective

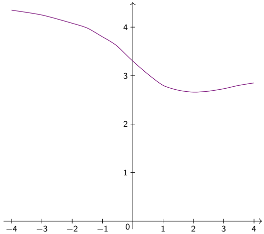
Inference goal

Compute an ideal value of θ that makes q_θ as close to p as possible, using an **optimization algorithm**

Application to the **inference problem**
two distributions over sampled variables z:

- $p(z|x)$:
  posterior probability distribution over z defined by the model (with the observation x)
- $q_\theta(z)$:
  guide probability distribution (parameterized by θ)

Plot of $\mathbf{D_{KL}}(q_\theta(z), p(z|x))$ as a function of θ:



Optimization objective: $\mathrm{argmin}_\theta \mathbf{D_{KL}}(q_\theta(z), p(z|x))$

Next step: achieve a **stochastic approximation of the gradient** of $\mathbf{D}_{\mathrm{KL}}$

Definition of the optimization objective

We seek for θ so as to minimize:

$$\begin{aligned}
&\mathbf{D}_{\mathrm{KL}}(q_\theta(z), p(z|x)) \\
&= \int_{\mathcal{Z}} q_\theta(z) \log \frac{q_\theta(z)}{p(z|x)} \mathrm{d}z \\
&= \int_{\mathcal{Z}} q_\theta(z) \log \frac{q_\theta(z)p(x)}{p(z,x)} \mathrm{d}z && \text{by def. of conditional prob} \\
&= \int_{\mathcal{Z}} q_\theta(z) \log p(x) \mathrm{d}z + \int_{\mathcal{Z}} q_\theta(z) \log \frac{q_\theta(z)}{p(z,x)} \mathrm{d}z \\
&= \log p(x) \cdot \int_{\mathcal{Z}} q_\theta(z) \mathrm{d}z + \int_{\mathcal{Z}} q_\theta(z) \log \frac{q_\theta(z)}{p(z,x)} \mathrm{d}z \\
&= \log p(x) \cdot 1 + \int_{\mathcal{Z}} q_\theta(z) \log \frac{q_\theta(z)}{p(z,x)} \mathrm{d}z && \text{as } q_\theta \text{ is a probability} \\
&= \log p(x) - \mathcal{L}(\theta)
\end{aligned}$$

where $\mathcal{L}(\theta)$ is called the ELBO:

Evidence Lower Bound (ELBO)

The **ELBO** is defined as $\mathcal{L}(\theta) = \int_{\mathcal{Z}} q_\theta(z) \log \frac{p(z,x)}{q_\theta(z)} \mathrm{d}z$.

Since $\mathbf{D}_{\mathrm{KL}}(q_\theta(z), p(z|x)) + \mathcal{L}(\theta) = \log p(x)$ and $\log p(x)$ does not depend in θ,

Minimizing $\mathbf{D}_{\mathrm{KL}}(q_\theta(z), p(z|x))$ is equivalent to maximizing $\mathcal{L}(\theta)$

Gradient of the ELBO... (1)

To perform **gradient ascent** in order to **maximize ELBO**, we first need to evaluate this gradient, and formulate it **as an expectation**:

$$\begin{aligned}
&\nabla_\theta \mathcal{L}(\theta) \\
&= \int_{\mathcal{Z}} \nabla_\theta \left[q_\theta(z) \log \frac{p(z,x)}{q_\theta(z)} \mathrm{d}z \right] \qquad \text{under assumption (later)} \\
&= \int_{\mathcal{Z}} (\nabla_\theta q_\theta(z)) \log \frac{p(z,x)}{q_\theta(z)} \mathrm{d}z + \int_{\mathcal{Z}} q_\theta(z) \nabla_\theta \left(\log \frac{p(z,x)}{q_\theta(z)} \right) \mathrm{d}z
\end{aligned}$$

Let us study the second term more:

$$\begin{aligned}
&\int_{\mathcal{Z}} q_\theta(z) \nabla_\theta \left(\log \frac{p(z,x)}{q_\theta(z)} \right) \mathrm{d}z \\
&= \int_{\mathcal{Z}} q_\theta(z) \nabla_\theta \left(\log p(z,x) - \log q_\theta(z) \right) \mathrm{d}z \\
&= \int_{\mathcal{Z}} q_\theta(z) \nabla_\theta \left(\log p(z,x) \right) \mathrm{d}z - \int_{\mathcal{Z}} q_\theta(z) \nabla_\theta \left(\log q_\theta(z) \right) \mathrm{d}z \\
&= - \int_{\mathcal{Z}} q_\theta(z) \nabla_\theta \left(\log q_\theta(z) \right) \mathrm{d}z \qquad \text{(the gradient of a constant is null)} \\
&= - \int_{\mathcal{Z}} q_\theta(z) \frac{\nabla_\theta q_\theta(z)}{q_\theta(z)} \mathrm{d}z \\
&= - \int_{\mathcal{Z}} \nabla_\theta q_\theta(z) \mathrm{d}z \\
&= -\nabla_\theta [\int_{\mathcal{Z}} q_\theta(z) \mathrm{d}z] \\
&= -\nabla_\theta 1 \qquad \text{(since } q_\theta \text{ defines a probability distribution)} \\
&= 0
\end{aligned}$$

Gradient of the ELBO... (2)

We have shown:

$$\nabla_\theta \mathcal{L}(\theta) = \int_{\mathcal{Z}} (\nabla_\theta q_\theta(z)) \log \frac{p(z,x)}{q_\theta(z)} \mathrm{d}z$$

We remark that $\nabla_\theta \log q_\theta(z) = \frac{\nabla_\theta q_\theta(z)}{q_\theta(z)}$

Thus, we can substitute $\nabla_\theta q_\theta(z) = q_\theta(z) \cdot \nabla_\theta \log q_\theta(z)$:

$$\begin{array}{rcl} \nabla_\theta \mathcal{L}(\theta) & = & \int_{\mathcal{Z}} (\nabla_\theta q_\theta(z)) \log \frac{p(z,x)}{q_\theta(z)} \mathrm{d}z \\ & = & \int_{\mathcal{Z}} q_\theta(z) \cdot \nabla_\theta \log q_\theta(z) \log \frac{p(z,x)}{q_\theta(z)} \mathrm{d}z \\ & = & \mathbb{E}_{q_\theta(z)} \left(\nabla_\theta \log q_\theta(z) \log \frac{p(z,x)}{q_\theta(z)} \right) \end{array}$$

This form, **as an expectation** is adapted to **stochastic approximation**
i.e., estimation based on a number of samples...

Stochastic approximation of the gradient of the ELBO

Based on $\nabla_\theta \mathcal{L}(\theta) = \mathbb{E}_{q_\theta(z)} \left(\nabla_\theta \log q_\theta(z) \log \frac{p(z,x)}{q_\theta(z)} \right)$ we may generate a sample ρ_0 from q_θ and produce

$$\mathbf{GrEst}_\theta(\rho_0) = (\nabla_\theta \log \mathcal{D}[\![q_\theta]\!](\mu_I, \rho_0)) \cdot \log \frac{\mathcal{D}[\![q_\theta]\!](\mu_I, \rho)}{\mathcal{D}[\![p]\!](\mu_I, \rho_0)}$$

Stochastic estimation repeats with **many samples**:

Stochastic approximant of gradient expectation formula

Using N samples from $\rho_0, \ldots, \rho_{N-1}$ from distribution q_θ:

$$\underline{\mathbf{GrEst}}_\theta(\rho) = \frac{1}{N} \sum_{i=0}^{N-1} \mathbf{GrEst}_\theta(\rho_i) = \frac{1}{N} \sum_{i=0}^{N-1} (\nabla_\theta \log \mathcal{D}[\![q_\theta]\!](\mu_I, \rho_i)) \cdot \log \frac{\mathcal{D}[\![q_\theta]\!](\mu_I,\rho_i)}{\mathcal{D}[\![p]\!](\mu_I,\rho_i)}$$

Implementation:

- sampling executions from q_θ, *e.g.*, by **rejection sampling**
- computation of $\mathcal{D}[\![q_\theta]\!]$ and $\mathcal{D}[\![p]\!]$ using **density semantics** based **instrumented implementation**; similarly, also accumulate $\nabla_\theta(\log \mathrm{pdf})$ (table for classical distributions)
- usually sum log pdf rather than multiplying pdf

SVI algorithm based on gradient optimization

Fixed parameters:

- N number of samples per iterate for stochastic estimation
- λ: **learning rate**, typically small, *e.g.*, $\lambda = 0.01$
- θ_{init}: **initial** value of the guide parameter (typically a rough guess)

Algorithm:

$$\left\{\begin{array}{l} \text{select } \theta_0 := \theta_{\mathrm{init}} \\ \textbf{repeat } K \text{ times} \\ \qquad \text{sample } r_0, \ldots, r_{N-1} \\ \qquad \theta_{k+1} \leftarrow \theta_k - \lambda \cdot \frac{1}{N} \cdot \displaystyle\sum_{i=0}^{N-1} \mathbf{GrEst}_{\theta_k}(r_i) \\ \textbf{produce } \theta_K \end{array}\right.$$

Properties (under assumptions discussed shortly)

- $\frac{1}{N} \cdot \sum_{i=0}^{N-1} \mathbf{GrEst}_{\theta_k}(\rho_i)$ provides an **unbiased** estimate of the gradient at θ_k
- the algorithm **converges to a local maximum** θ of $\mathcal{L}$, *i.e.*, a **local minimum** of $\mathbf{D}_{\mathrm{KL}}$...

Outline

Another model-guide pair

Model:

```
def model(...):
    ...
    sigma = pyro.sample("sigma", Uniform(0., 10.))
    ...
    pyro.sample("obs", Normal(..., sigma), obs=...)
```

Guide:

```
def guide(...):
    ...
    loc = pyro.param("sigma_loc", 1., constraint=constraints.positive)
    ...
    sigma = pyro.sample("sigma", Normal(loc, 0.05))
```

Issue:

- domain of `sigma` in the **model:** $[0, 10]$
- domain of `sigma` in the **guide:** $\mathbb{R}$
- thus, **KL-divergence is undefined**

(Example taken from the **Pyro webpage examples**...)

Issues possibly leading to undefinedness of KL-divergence

Absolute continuity requirement:

- definition of KL-divergence:

$$\mathbf{D}_{\mathrm{KL}}(q_\theta, p) = \mathbb{E}_{q_\theta}\left(\log \frac{q_\theta}{p}\right) = \int_{\mathcal{X}} q_\theta(\mathrm{d}x) \log \frac{q_\theta(\mathrm{d}x)}{p(\mathrm{d}x)}$$

- absolute continuity requirement: model distribution p and guide distribution q should have **the same zero probability regions**
  otherwise: **KL divergence is undefined**
- domain in **model** $[0, 10]$, in **guide** $\mathbb{R}$
  leads to **the violation of absolute continuity assumption**
  e.g., and **KL divergence is undefined**

Anther possible issue: integrability

- $q_\theta(\mathrm{d}x) \log \frac{q_\theta(\mathrm{d}x)}{p(\mathrm{d}x)}$ may not be integrable
  ...even when absolute continuity holds

Our goal: define **semantics** to let **static analysis** provide guarantees

Informal overview of potential SVI issues

Several assumptions are necessary:

- **KL-divergence must be defined, not ∞:**
  otherwise: undefined optimization objective
- **KL-divergence must be differentiable:**
  otherwise: incorrect gradient descent
- **the stochastic estimate of $\nabla \mathbf{D_{KL}}(q_\theta, p)$ should be well-defined, and unbiased:**
  otherwise: incorrect computation of gradient descent approximation

Practical consequences are difficult to troubleshoot, *e.g.*,

- crashes or divergence of the inference engine
- incoherent / invalid optimization results
  may be very difficult to even notice

Unbiasedness conditions

Sufficient conditions (unbiasedness may hold in some cases where assumptions are violated, especially if locally so):

Theorem: unbiasedness of gradient estimate of KL divergence

If:

1. absolute continuity: $\mathcal{D}[\![D_\theta]\!](\mu_I)(\rho) \Longrightarrow \mathcal{D}[\![C]\!](\mu_I)(\rho)$
2. differentiability: $\theta \mapsto \mathcal{D}[\![D_\theta]\!](\mu_I)(\rho)$ differentiable wrt all components
3. boundnedness of KL divergence
4. differentiability of KL divergence wrt all its arguments
5. integral permutation conditions on KL divergence and guide density $\int \nabla \ldots = \nabla \int \ldots$

Then:

$$\mathbb{E}(\nabla_\theta \mathbf{D}_{\mathrm{KL}}(\mathcal{D}[\![D_\theta]\!](\mu_I), \mathcal{D}[\![C]\!](\mu_I))) \equiv \frac{1}{N} \cdot \sum_{i=0}^{N-1} \mathbf{GrEst}_{\theta_k}(\rho_i)$$

Full version:

Towards Verified Stochastic Variational Inference for Probabilistic Programs
Wonyeol Lee, Hangyeol Yu, Xavier Rival and Hongseok Yang
POPL'20

Outline

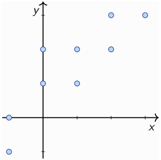
Abstract interpretation primer: state abstraction

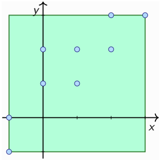
Static analysis: interpretation using an **abstract domain**

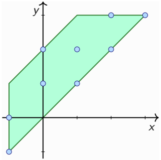
Abstract domains (Cousot & Cousot, 1977)

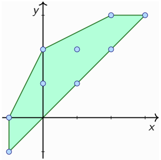
- Families of **abstract predicates** adapted to static analysis
- **Compact** and **efficient** representations
- **Operations** for the static analysis of concrete operations
- **Mapping from abstract to concrete**: **concretization function** γ

Ex., numeric abstractions: abstraction of **sets of pairs of integers**









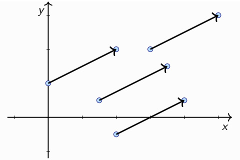
Abstract states **over-approximate** sets of concrete states

Abstract interpretation primer: analysis of post-conditions

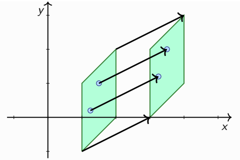
Computing sound abstract transformer

- **Conservative analysis** of concrete execution steps in the abstract
  e.g., **assignments**, **condition tests**...
- May **lose precision**, will **never forget any behavior**
- Balance between **cost** and **precision**

Example: analysis of a **translation** with **octagons** (**exact**!)



concrete transformation



abstract transformation

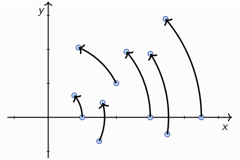
Soundness: all concrete behaviors are taken into account

Abstract interpretation primer: analysis of post-conditions

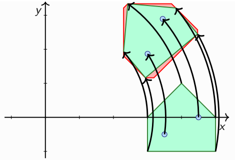
Computing sound abstract transformer

- **Conservative analysis** of concrete execution steps in the abstract *e.g.*, **assignments**, **condition tests**...
- May **lose precision**, will **never forget any behavior**
- Balance between **cost** and **precision**

Example: analysis of a **40 deg. rotation** with **octagons** (**approximate**!)



concrete transformation



abstract transformation

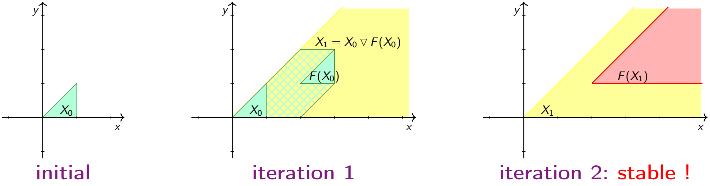
Soundness: all concrete behaviors are taken into account

Abstract interpretation primer: analysis of loops

Computing invariants about infinite executions with widening ∇

- **Widening** ∇ over-approximates $\cup$: **soundness guarantee**
- **Widening** ∇ guarantees the **termination of the analyses**
- Typical choice of ∇: **remove unstable constraints**

Example: iteration of the translation $(2, 1)$, with **octagons**



Soundness: all concrete behaviors are taken into account

A generic static analysis

We set up a static analysis, **parameterized by an abstract domain:**
Logical predicates + **representation** + **algorithms**

Abstract domain

An **abstract domain** comprises a set of abstract predicates $\mathbb{D}^\sharp$ and:

- **concretization function** $\gamma : \mathbb{D}^\sharp \to \mathbb{D}$
  where $\mathbb{D} = \mathbb{S} \to \mathcal{P}(\mathbb{S} \times \mathbb{R}^+ \times \mathbb{R}^+ \uplus \{\perp\})$
- **least element** $\perp$ with $\gamma(\perp) = \emptyset$
- **widening operator** $\nabla : \mathbb{D}^\sharp \times \mathbb{D}^\sharp \longrightarrow \mathbb{D}^\sharp$
  over-approximating $\cup$
  and enforcing termination on all sequences of abstract iterates
- **abstract composition** $comp^\sharp : \mathbb{D}^\sharp \times \mathbb{D}^\sharp \longrightarrow \mathbb{D}^\sharp$
  soundness: $\forall g_0 \in \gamma(d_0^\sharp), \forall g_1 \in \gamma(d_1^\sharp),\ (g_0 \circ g_1) \in \gamma(comp^\sharp(d_0^\sharp, d_1^\sharp))$
- **abstract conditions, assignment, sample and score operations**
  satisfying similar soundness conditions

Abstract interpreter

The definition of the abstract interpreter follows by induction over the syntax:

The abstract interpreter

$$
\begin{array}{rcl}
[\![\mathrm{skip}]\!]^\sharp & \triangleq & \mathit{skip}^\sharp \\
[\![\mathrm{if}\ E\ \{C_0\}\ \mathrm{else}\ \{C_1\}]\!]^\sharp & \triangleq & \mathit{cond}^\sharp(E)([\![C_0]\!]^\sharp, [\![C_1]\!]^\sharp) \\
[\![x := E]\!]^\sharp & \triangleq & \mathit{assign}^\sharp(x, E) \\
[\![x := \mathrm{sample}_{\mathcal{N}}(k, E_1, E_2)]\!]^\sharp & \triangleq & \mathit{sample}^\sharp(x, k, E_1, E_2) \\
[\![C_0; C_1]\!]^\sharp & \triangleq & \mathit{comp}^\sharp([\![C_1]\!]^\sharp, [\![C_0]\!]^\sharp) \\
[\![\mathrm{observe}_{\mathcal{N}}(E_0, E_1, E_2)]\!]^\sharp & \triangleq & \mathit{score}^\sharp(E_0, E_1, E_2) \\
[\![\mathrm{while}\ E\ \{C\}]\!]^\sharp & \triangleq & \mathbf{lfp}(\lambda d^\sharp \cdot \mathit{cond}^\sharp(E)(\mathit{comp}^\sharp(d^\sharp, [\![C]\!]^\sharp), \mathit{skip}^\sharp))
\end{array}
$$

Exercise: provide the soundness condition for each of the operations

Soundness

The abstract semantics is sound in the sense that it **over-approximates** the effect of program concrete executions, in the sense of the density semantics:

Theorem: static analysis soundness

For all command C:

$$[\![C]\!]_{\mathcal{D}} \in \gamma([\![C]\!]^{\sharp})$$

Proof: **exercise!**

The abstract semantics is not complete, and may not return the most precise abstraction for a given program...

First instance: static analysis for model/guide support match

Goal: **discharge model/guide support equality** (absolute continuity)

Abstraction

We define $\mathbb{D}^\sharp$ and γ by $\mathbb{D}^\sharp = \{\bot^\sharp, \top^\sharp\} \uplus \mathcal{P}(\mathbb{K})$ and:

$$
\begin{array}{llll}
\gamma: & \bot^\sharp & \longmapsto & \lambda((\mu,\rho), w, p)\cdot\bot \\
 & \top^\sharp & \longmapsto & \mathbb{D} \\
 & K(\subseteq \mathbb{K}) & \longmapsto & \{g \in \mathbb{D} \mid \\
 & & & \quad [\forall((\mu,\rho), w, p), \mu', w', p' \\
 & & & \quad\quad g((\mu,\rho), w, p) = (\mu', \emptyset, w', p') \Longrightarrow \mathrm{Dom}(\rho) = K]\}
\end{array}
$$

A few transfer functions:

$$
\begin{array}{rcl}
\mathit{comp}^\sharp(\bot^\sharp, d^\sharp) = \mathit{comp}^\sharp(d^\sharp, \bot^\sharp) & = & \bot^\sharp \\
\mathit{comp}^\sharp(\top^\sharp, d^\sharp) = \mathit{comp}^\sharp(d^\sharp, \top^\sharp) & = & \top^\sharp \\
\mathit{comp}^\sharp(K_0, K_1) & = & \left\{ \begin{array}{ll} K_0 \uplus K_1 & \text{if } K_0 \cap K_1 = \emptyset \\ \top^\sharp & \text{otherwise} \end{array} \right. \\
\mathit{sample}^\sharp(x, k, E_0, E_1) & = & \{k\} \\
\mathit{score}^\sharp(x, E_0, E_1, E_2) & = & \emptyset
\end{array}
$$

First instance: static analysis for model/guide support match

Goal: **discharge model/guide support equality** (absolute continuity)

Abstraction

We define $\mathbb{D}^\sharp$ and γ by $\mathbb{D}^\sharp = \{\bot^\sharp, \top^\sharp\} \uplus \mathcal{P}(\mathbb{K})$ and:

$$\begin{array}{rlcl} \gamma: & \bot^\sharp & \longmapsto & \lambda((\mu,\rho),w,p)\cdot\bot \\ & \top^\sharp & \longmapsto & \mathbb{D} \\ & K(\subseteq \mathbb{K}) & \longmapsto & \{g \in \mathbb{D} \mid \\ & & & \quad [\forall((\mu,\rho),w,p),\mu',w',p' \\ & & & \quad\quad g((\mu,\rho),w,p) = (\mu',\emptyset,w',p') \Longrightarrow \mathrm{Dom}(\rho) = K]\} \end{array}$$

Example analysis:

```
def model():
  x = pyro.sample("v", Normal(0., 5.))
  if (x > 0):
    pyro.sample("obs", Normal(1., 1.), obs=x)
  else:
    pyro.sample("obs", Normal(−2., 1.), obs=x)
```

Then: $[\![\texttt{model}]\!]^\sharp = \{\texttt{v}\}$

First instance: static analysis for model/guide support match

Goal: **discharge model/guide support equality** (absolute continuity)

Abstraction

We define $\mathbb{D}^\sharp$ and γ by $\mathbb{D}^\sharp = \{\bot^\sharp, \top^\sharp\} \uplus \mathcal{P}(\mathbb{K})$ and:

$$\begin{array}{llcl}
\gamma: & \bot^\sharp & \longmapsto & \lambda((\mu,\rho),w,p)\cdot\bot \\
 & \top^\sharp & \longmapsto & \mathbb{D} \\
 & K(\subseteq \mathbb{K}) & \longmapsto & \{g \in \mathbb{D} \mid \\
 & & & \quad [\forall((\mu,\rho),w,p),\mu',w',p' \\
 & & & \qquad g((\mu,\rho),w,p) = (\mu',\emptyset,w',p') \Longrightarrow \mathrm{Dom}(\rho) = K]\}
\end{array}$$

Generalization: case where we consider **many distributions** (and not only normal distributions)

- the basic abstraction will not work:
  sampling k from a normal distribution and sampling k from a Bernoulli distribution yield **distinct supports**
- new abstraction: $\mathbb{K} \to$ **Distributions** $\uplus \{\bot, \top\}$

Second instance: static analysis for guide differentiability

Goal: **discharge differentiability properties**

Exercise:

- abstraction choice
- transfer functions

Outline

Conclusion

Main ideas to remember from this lecture:

- SVI turns an inference problem into an **optimization problem**, to select an (possibly) optimal program in a parameterized family
- optimization only works when **a series of assumptions hold**

A few suggestions for **reading**:

- on **probabilistic semantics**:
  Dexter Kozen,
  Semantics of Probabilistic Programs.
  Journal of Computing Systems Science(1981)
- on **SVI**, among other inference techniques:
  Jan-Willem van de Meent, Brooks Paige, Hongseok Yang, Frank Wood,
  An Introduction to Probabilistic Programming.
  (2018)
- on **correctness of SVI**:
  Wonyeol Lee, Hangyeol Yu, Xavier Rival, Hongseok Yang,
  Towards verified stochastic variational inference for probabilistic programs,
  PACM-POPL (2020)